УДК 681.3

В.А. Мар'яновський, О.В. Дворцов, Ю.В. Бойко, Д.Б. Грязнов, С.Д. Погорілий

ПІДХІД ДО РЕАЛІЗАЦІЇ ІНТЕРФЕЙСІВ КЛАСТЕРНИХ СИСТЕМ

Виконано порівняльний аналіз методів реалізації інтерфейсів для кластерних систем, побудованих на платформі MS HPC, внаслідок чого обґрунтовано метод формування Web-інтерфейсу такого кластера. Відповідно до запропонованого методу наведено варіант реалізації інтерфейсу кластера, впровадженого в MS IT-академії інформаційно-обчислювального центру Київського національного університету імені Тараса Шевченка.

Вступ

Одним із основоположних принципів відкритих систем є доступність (availability), що висуває задачу наявності гнучкого та зручного доступу до потужних обчислювальних ресурсів таких систем. Вона досягається при використанні універсального механізму доступу користувачів до обчислювальних систем з урахуванням безпеки та гнучкості при користуванні.

Існує фізичне обмеження на збільшення тактової частоти мікропроцесорів комп'ютерів, тому альтернативою для збільшення обчислювальної потужності є використання паралелізму при розрахунках. Поява кластерних систем та паралельних обчислень дозволяє значно підвищити масштаб застосування потужних комп'ютерів [1].

У статті розглянуто методи доступу до кластерних обчислювальних систем та виконано їх порівняльний аналіз. Запропоновано концепцію створення інструментальних засобів для роботи з кластером, що забезпечує досить гнучку роботу користувача в поєднанні з забезпеченням відповідними функціями та повним контролем його дій. Підхід орієнтовано на використання кластерів побудованих на програмній платформі Windows HPC Server 2008 (HPC) для 64-х розрядної операційної системи (ОС) Windows Server 2008 [2]. До складу HPC входить підтримка Microsoft MPI (MS MPI) – стек MPI, оснований на стандарті MPICH2 Аргонської національної лабораторії та повністю з ним сумісний, а також можливе використання інших стеків MPI, які розроблені відповідно до стандарту MPI2 (MPI2 – це розширення вихідної специфікації MPI).

Один з прикладів кластера, розглянутого в роботі, реалізовано в Київському національному університеті імені Тараса Шевченка [3]. У ролі комунікаційного середовища між вузлами кластера використано протокол Ethernet зі швидкістю передачі даних 1 Гбіт. Такий варіант комунікаційного середовища є найбільш розповсюдженим та задовольняє вимогам побудови кластерних систем, що було показано в [4]. Вибір швидкості передачі комунікаційного середовища обумовлений оптимальним співвідношення між затримками при передачі даних та вартістю обладнання.

Методи доступу до кластерних систем

Незалежно від методу побудови кластера завжди необхідно передбачити метод доступу користувачів до обчислювальних ресурсів. Один з методів полягає у виділенні окремого комп'ютера для запуску задач на кластері та доступ до результатів виконаних обчислень. Характерними особливостями методу є:

- необхідність виділення окремого робочого місця для користувачів кластера;
- контрольованість доступу до обчислювальних ресурсів;
- ускладнена організація доступу до кластера, враховуючи, що в загальному випадку робоче місце має бути розташоване в спільній з кластером локальній мережі. Найбільш гостро ця проблема постає при користуванні кластером співробітниками сторонніх організацій.

Зазначені недоліки стають більш актуальними для кластерних систем у ве-

ликих або децентралізованих організаціях. Тому, альтернативним методом є дистанційний доступ до кластера, який можливо реалізувати декількома варіантами:

- на основі протоколів дистанційного доступу;
- використання вбудованої графічної компоненти для роботи з задачами на кластері – Compute Cluster Job Manager (CCJM);
- створення Web-інтерфейсу.

При побудові кластерних систем на базі інших платформ та ОС (не Microsoft), можливі інші методи, пов'язані з особливостями платформ та ОС, які будуть використані.

У ролі програмного забезпечення для дистанційного доступу до кластеру може виступати:

- вбудований в ОС сервіс термінальних підключень на основі протоколу RDP (Remote Desktop Protocol), клієнтська частина якого вбудована в майже всі версій ОС Windows, Linux, FreeBSD, Mac OS X;
- Radmin (програмне забезпечення без можливості вільного розповсюдження), яке дозволяє отримати доступ до віддаленого ресурсу з відсутністю можливості одночасної роботи декількох користувачів. Недоліком виступає необхідність попередньої інсталяції клієнтської та серверної частин програмного забезпечення, що зменшує доступність кластерної системи в загальному випадку;
- RealVNC – програмне забезпечення, функціонально аналогічне Radmin.

На початкових етапах роботи кластера в Київському національному університеті імені Тараса Шевченка доступ було реалізовано з використанням досить розповсюдженого протоколу RDP. За функціональністю дистанційний доступ не відрізняється від виділення окремого робочого місця для роботи з кластером, також його перевагою є можливість функціонування через низькошвидкісне з'єднання. З іншого боку, використовуючи цей метод, користувач отримує доступ до термінального сервера з можливостями, які обмежені лише груповими політиками, тому задача обмежень прав доступу та проведення облік дії користувача досить ускладнена, що зменшує рівень захищеності системи.

Ще один метод дистанційного доступу до кластера користувачем полягає у використанні CCJM, для цього необхідно встановити програмне забезпечення CCJM на комп'ютер користувача та використовувати його для встановлення та контролю виконання задач на кластері. В загальному випадку комп'ютер користувача має бути в одному домені з кластером під керуванням служби Active Directory (AD), а отже доступ до кластера в такому режимі можливий лише в межах мережі однієї організації або одного підрозділу, що накладає обмеження на його використання, а також на клієнтську ОС. Це пов'язано з відсутністю у CCJM функціональності пов'язаної з аутентифікацією користувачів, тому вона має проходити використовуючи вбудовані можливості клієнтської операційної системи, що виконує служба AD.

Всі функціональні можливості, необхідні користувачу, можна реалізувати за допомогою Web-інтерфейсу, що дозволяє надати йому найбільш універсальний інструмент дистанційної роботи. У цьому методі немає обмежень на кількість одночасно працюючих користувачів у порівнянні з попередніми методами і він не накладає обмежень на клієнтську ОС. Це метод також забезпечує найбільш високий рівень захищеності кластера від користувача і дозволяє повністю контролювати всі його дії. Ураховуючи такі переваги доступ до кластера в університеті було замінено на використання Web-інтерфейсу.

Створення інтерфейсу доступу до обчислювальних ресурсів

Для реалізації Web-інтерфейсу можливо використовувати різноманітні інструментальні засоби в різних середовищах програмування, але враховуючи, що він призначений для забезпечення доступу до Windows платформ, найбільша сумісність досягається при використані інструментарію компанії Microsoft. Тому, інтерфейс створювався з використанням мови програмування C# у середовищі розробки Microsoft Visual Studio 2008 на платформі ASP.NET. Як Web-сервер використано

Internet Information Services (IIS) на ОС Windows Server 2008.

Базові функціональні можливості інтерфейсу дозволяють здійснювати:

- аутентифікацію користувачів;
- завантаження файлів (вхідних даних для задач, які виконуються на кластері);
- роботу з результатами кластерних обчислень;
- встановлення та моніторинг задач на кластері;
- збір різноманітних статистичних даних роботи кластера та його завантаженості.

Будь-яка інша функціональність є додатковою у залежності від специфіки реалізації обчислювального ресурсу в різноманітних організаціях. Додатковими функціональними можливостями можуть бути:

- компіляція вихідного коду задач;
- механізм обміну повідомленнями між користувачами;
- обмін досвідом по користуванню кластером;
- читання новин;
- тощо.

Обчислювальні ресурси об'єднуються в єдину систему та мають централізоване керування на основі служби AD, під керуванням якої знаходяться і користувачі кластера. Тому найбільш коректний метод аутентифікації користувачів має бути побудований на роботі з службою AD. Для цього може бути використано протокол LDAP (Lightweight Directory Access Protocol), який підтримується технологією ASP.NET. Є декілька методів роботи з AD, що ґрунтуються на протоколі LDAP. Один з них може бути реалізований з використанням простору імен System.DirectoryServices. Для перевірки користувача спочатку необхідно ініціалізувати об'єкт типу DirectoryEntry, зазначений при цьому параметри облікового запису адміністратора (користувача з правами роботи з службами AD) та шлях до AD-серверу, який складається, у свою чергу з протоколу (LDAP) та адреси сервера. Потім за допомогою методу FindAll() об'єкта типу DirectorySearch перевірити наявність необхідного користувача в службі AD. Мовою програмування C# фрагмет коду виглядає так:

```
using System.DirectoryServices;
// ...
// Звернення до AD під обліковим записом адміністратора з відповідним
// іменем та паролем до AD домену (domain.name)
DirectoryEntry de = new DirectoryEntry(    "LDAP://domain.name",
                                           "_admin_username",
                                           "_admin_password",
                                           AuthenticationTypes.Secure);
de.username = “user_name”;
de.password = “user_password”;
// Пошук по AD
DirectorySearch deSerach = new DirectorySearch();
deSearch.SearchRoot = de;
// Визначення кількості знайдених результатів
if( deSearch.FindAll().Count > 0 )
        // Ім’я та пароль користувача вірний
else
        // Помилкові параметри користувача при аутентифікації
```

Для платформи ASP.NET використовується конфігураційний файл web.config, який може бути визначеним для кожного каталогу. В конфігураційному файлі слід зазначити, що для доступу до вмісту каталогу необхідно пройти аутентифікацію, а також зазначити сторінку, яка буде використана для цього. Фрагмент коду буде виглядати наступним чином (конфігураційний файл задається мовою XML):

```
<authentication mode="Forms">
        <!—- Login.aspx – сторінка для аутентифікації -->
        <!—- Default.aspx – перша сторінка після аутентифікації -->
    <forms name="ADAuth" timeout="10"
         loginUrl="Login.aspx" defaultUrl="Default.aspx">
    </forms>
</authentication>
<authorization>
        <!—- заборона доступу невідомому користувачу -->
        <deny users="?"/>
        <allow users="*"/>
</authorization>
```

Для встановлення та моніторингу задач на кластері використано простір імен Microsoft.ComputeCluster, для його використання необхідний набір інструментальних засобів для розробки (Software Development Kit [5]), який входить до Compute Cluster Pack. Робота з задачами на кластері вимагає комбінації імені облікового запису та пароля у службі AD, це ще одна причина, з якої аутентифікація користувачів кластера має бути інтегрованою зі службою AD, її необхідно виконати до початку роботи з кластером. Для отримання набору задач на кластері використовується метод ListAllJobs(), який повертає набір задач на кластері в об’єкті типу IEnumerator. Для проходу по всьому набору задач використовується метод MoveNext() у циклі while, а поточну задачу можна отримати використовуючи властивість Current. Фрагмент коду мовою C# виглядає наступним чином:

```
using Microsoft.ComputeCluster;
// ...
// 192.168.1.1 – ip-адреса головного вузла кластеру
ICluster cluster = new Cluster();
cluster.Connect("192.168.1.1");
// Отримання списку задач на кластері
IEnumerator jobsList = cluster.ListAllJobs().GetEnumerator();
IJob currentJob;
jobsList.Reset();
// Прохід по всім задачам на кластері
while (jobsList.MoveNext())
{
    currentJob = (IJob)jobsList.Current;
    // Відображення певним чином currentJob користувачу.
}
```

Запропонований метод формування графічного інтерфейсу дозволяє сформувати вікно, яке показано на рис. 1.

Роботу з вихідними даними та встановлення початкових даних задач кластера можна виконувати використовуючи файли. Кожен з вузлів кластера повинен мати доступ до вхідних і вихідних даних. Для забезпечення спільної роботи з файлами в кластері було використано протокол Windows Server Message Block (SMB) Protocol, тому в інтерфейсі робота з результатами зводиться до роботи з файлами за протоколом SMB.

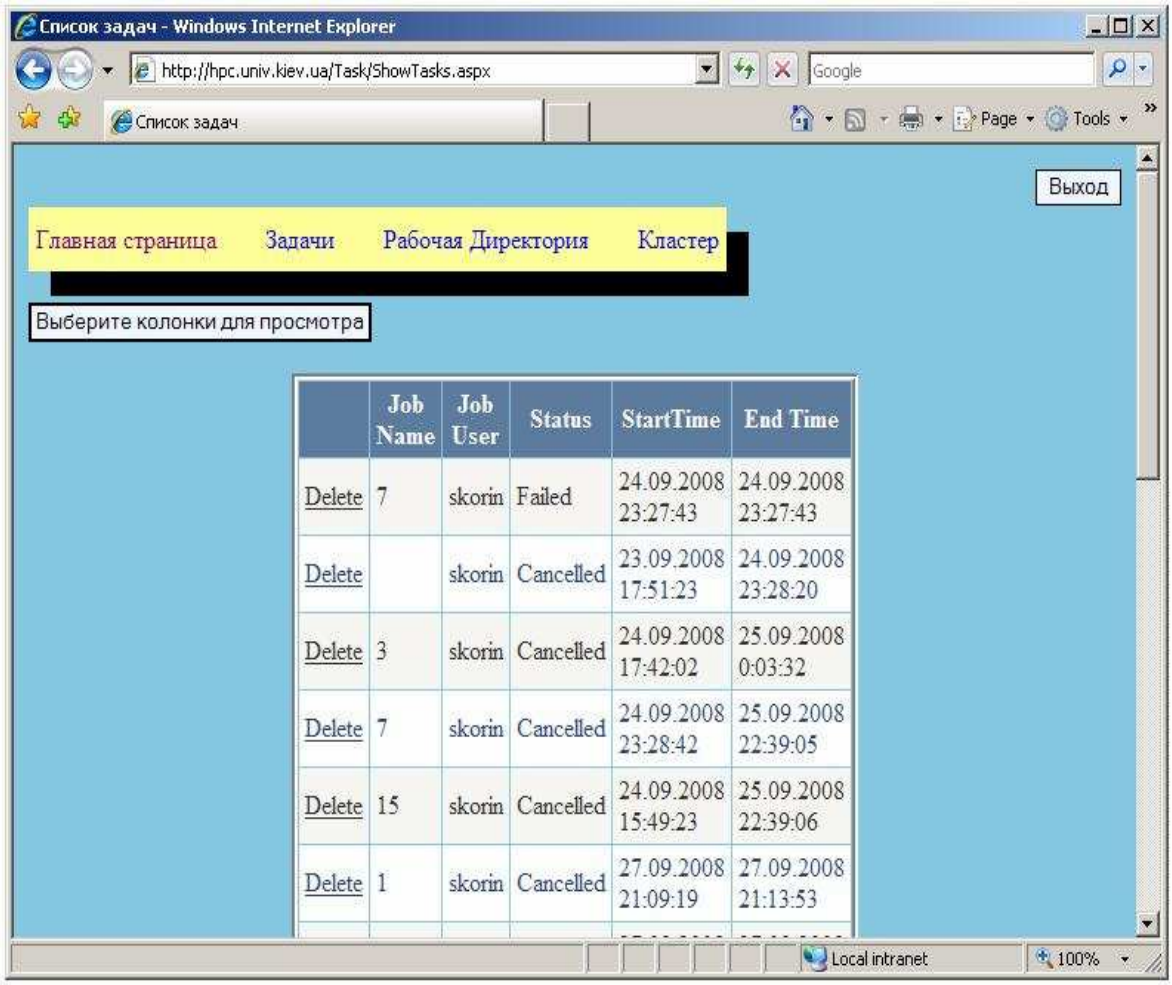

Рис. 1. Список задач на кластері

Одна з найважливіших функцій інтерфейсу полягає у встановленні задач на кластер. У термінології HPC для встановлення задачі на кластер необхідно створити проект (job), зазначивши його назву та мінімальну кількість процесорів при якому буде розпочате виконання проекту та максимальну кількість процесорів, яку може зайняти проект. Проект у свою чергу може складатися з довільної кількості завдань (task). Для створення завдання задаються такі базові параметри: мінімальна та максимальна кількість процесів для завдання, назва, робоча директорія (робота з файлами відбувається за протоком SMB), файли для стандартного потоку вводу-виводу, файл помилок та команда для виконання завдання. Враховуючи той факт, що програма користувача (pi.exe) розроблена з використанням інтерфейсу MPI, вона розпочинається з команди mpiexec. Після створення необхідної кількості завдань проект можна надіслати на виконання. Код фрагмента мовою C# виглядає так:

```
using Microsoft.ComputeCluster;
// ...
ClusterConnect cluster_connect = ClusterConnect.GetInstance();
// Перед встановленням задач на кластер, необхідно встановити з’єднання
// з кластером, яке було виконано попередньо
// ...
IJob m_job = new Job();
ITask task = new Microsoft.ComputeCluster.Task();
// Поділювальний каталог використовується як робочий
task.WorkDirectory = "\\user\bin\";
task.CommandLine = "mpiexec pi.exe 1000";
// Стандартний файл вводу, виводу та помилок
task.Stdin = "stdin.txt";
task.Stdout = "stdout.txt";
task.Stderr = "stderror.txt";
// Мінімально та максимально необхідна кількість процесорів для завдання
task.MinimumNumberOfProcessors = 2;
task.MaximumNumberOfProcessors = 4;

// Процедура додавання завдання до проекту
m_job.AddTask(task);
m_job.Name = “Job name”;
m_job.Project = “Project name”;
// Необхідна мінімальна та максимальна кількість процесорів для проекту
m_job.MinimumNumberOfProcessors = 2;
m_job.MaximumNumberOfProcessors = 4;

// Встановлення та запуск проекту на кластері
cluster_connect.cluster.AddJob(m_job);
// Властивості username та password для об’єкту cluster_connect
// були вказані користувачем при аутентифікації
cluster_connect.cluster.SubmitJob(   m_job.Id, cluster_connect.username,
                                     cluster_connect.password, true,0);
```

Збір статистики функціонування кластера: кількість встановлених задач на кластері (рис. 2), завантаженість кластера в певний момент часу, час виконання задач тощо. Все це можливо реалізувати використовуючи простір імен Microsoft. ComputeCluster або розробити персональну службу, яка буде збирати періодично необхідну інформацію з вузлів кластера. Для роботи HPC використовує базу даних Microsoft SQL Server, тому збір інформації про кластер можливо виконати через звернення до його бази даних декількома методами:

- використовуючи мову структурованих запитів SQL (Structured Query Language), яка використовується для створення, модифікації та керування даними в реляційних базах даних;
- використовуючи спеціальну мову запитів для .Net Framewaork – LINQ (Language Integrated Query), що дозволяє виконувати запити до типізованих даних, об’єктів які знаходяться в пам'ять чи в XML документі.

Мова LINQ, враховуючи її універсальність та простоту використання, була обрана для роботи з базою даних. Для збору статистики кластера використовуючи мову LINQ необхідно в конфігураційному файлі (web.config) задати параметри з’єднання з базою даних кластера.

Код конфігураційного файлу мовою XML, який формує рядок з'єднання (CCPClusterServiceConnectionString) до бази даних (CCPClusterService), виглядає так:

```
<connectionStrings>
        <add name="CCPClusterServiceConnectionString" connectionString="Data
        Source=HEADNODE\COMPUTECLUSTER;Initial
                Catalog=CCPClusterService;Integrated Security=True"
                providerName="System.Data.SqlClient" />
 </connectionStrings>
```

Розглянемо приклад звернення до таблиці ClusterDiagnostics, використовуючи мову запитів LINQ для отримання списку вузлів кластера (фрагмент коду сформовано мовою C#):

```
ClusterDiagnosticsDataContext DB_Linq = new ClusterDiagnosticsDataContext();
List<NodeDiagnostics> nodes_list;
nodes_list = (
from diag in DB_Linq.ClusterDiagnostics
    group diag by diag.NodeName into nodes
    select new NodeDiagnostics(nodes.FirstOrDefault().NodeName)).ToList();
foreach (NodeDiagnostics node in nodes_list)
{
        //проходження по всім вузлам кластера присутні в ClusterDiagnostics
}
```

У цьому прикладі типи ClusterDiagnosticsDataContext та NodeDiagnostics мають бути попередньо визначеними і відповідати структурі існуючої бази даних, в якій зберігається статистика роботи кластера (відповідає типу ClusterDiagnosticsDataContext). Властивість ClusterDiagnostics об'єкта DB_Linq відповідає таблиці ClusterDiagnostics у базі, а NodeDiagnostics інформації з таблиці ClusterDiagnostics, яка стосується кожного вузла кластера окремо.

Однією з додаткових можливостей може буди компіляція вихідного коду в виконуваний файл. Для цього необхідно надати можливість користувачу завантажувати вихідні файли на кластер та надати можливість компіляції, яка в такому разі буде виконуватись через надсилання відповідних дистанційних команд, тобто з використанням інтерфейсу командного рядка. При використанні HPC відсутня необхідність компіляції вихідного коду безпосередньо на вузлах кластера, що дає змогу користувачу використовувати попередньо відкомпільований та відлагоджений виконуваний файл. У ролі найбільш зручного та потужного інструмента для розробки користувачами може бути використано Visual Studio, яку необхідно буде встановити кожному з користувачів окремо. Для кластера побудованого на платформі Unix, необхідна компіляція вихідного коду безпосередньо на вузлах кластера [6]. Тому, при створенні Web-інтерфейсу для кластерів побудованих на інших платформах необхідність компіляції вихідного коду має бути обов'язковою умовою.

Розглянутий інтерфейс користувача повністю зворотно сумісний з попередньою версією MS HPC – Microsoft Compute Cluster Server 2003.

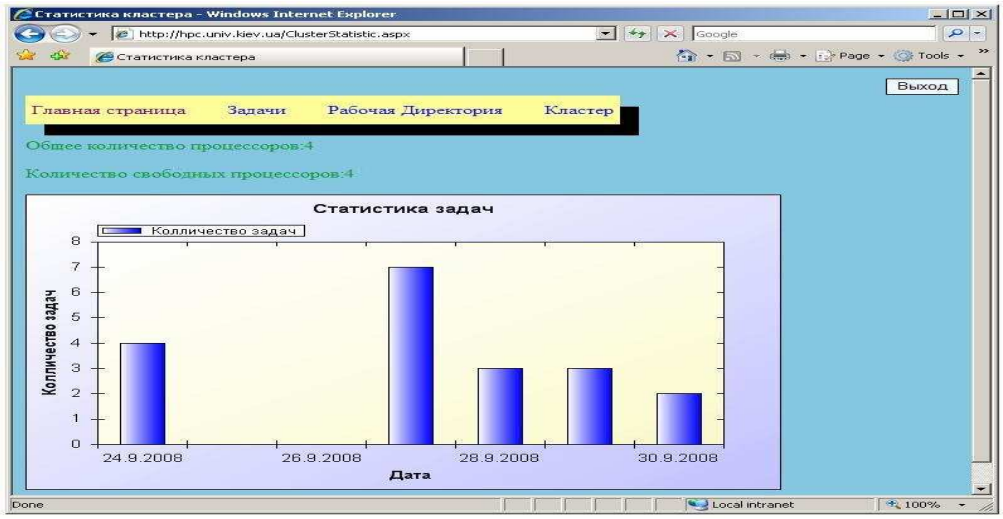

Рис. 2. Статистика встановлення задач

Висновки

У роботі проаналізовано методи реалізації інтерфейсів кластерів, внаслідок чого запропоновано методи побудови інтерфейсу кластера з використанням HPC. Виконано порівняльну характеристику запропонованих методів. Сформовано підхід до побудови програмних засобів в організації для доступу до її потужного обчислювального ресурсу. Запропонований метод вже реалізовано експериментально і він функціонує в обчислювальному центі Київського національного університету імені Тараса Шевченка.

1. *Антонов А.С.* Параллельное программирование с использованием технологий MPI // Изд-во Моск. ун-та. – 2004. –С. 5.
2. [*Інтернет* посилання]: Windows HPC Server 2008 http://www.microsoft.com/hpc/default.aspx
3. [*Інтернет* посилання]: Програмне забезпечення UACluster http://codeplex.com/UACluster
4. *Погорілий С.Д., Бойко Ю.В., Грязнов Д.Б.,* та інші. Концепція створення гнучких гомогенних архітектур кластерних систем // Проблеми програмування. – 2008. – № 2-3. – С. 84–90.
5. [*Інтернет* посилання]: Microsoft Compute Cluster SDK http://www.microsoft.com/
6. [*Інтернет* посилання]: Обчислювальний кластер Київського національного університету імені Тараса Шевченка http://cluster.univ.kiev.ua/

Отримано 16.01.2009

Про авторів:

Марьяновський Віталій Анатолійович, аспірант кафедри напівпровідникової електроніки, Київського національного університету імені Тараса Шевченка,

Дворцов Олександр Валерійович, студент радіофізичного факультету, Київського національного університету імені Тараса Шевченка,

Бойко Юрій Володимирович, завідувач кафедри напівпровідникової електроніки, Київського національного університету імені Тараса Шевченка,

Грязнов Дмитро Борисович, асистент кафедри напівпровідникової електроніки, Київського національного університету імені Тараса Шевченка,

Погорілий Сергій Дем'янович, професор кафедри напівпровідникової електроніки, Київського національного університету імені Тараса Шевченка.

Місце роботи авторів:

Київський національний університет імені Тараса Шевченка,
Тел.: 521 3347,
vitalik_m@univ.kiev.ua, dvortsov@ukr.net, boyko@univ.net.ua, dima@univ.kiev.ua, sdp@rpd.univ.kiev.ua